Build/Release Planning Guidelines

Number: 580-GL-076-01
Effective Date: October 1, 2009
Expiration Date: October 1, 2014

Approved By: (signature)
Name: John Donohue
Title: Chief, SED

Responsible Office: 580/Software Engineering Division (SED)
Title: Build/Release Planning Guidelines

Asset Type: Guideline
PAL Number: 1.2.5.1

Purpose

The purpose of this document is to provide uniform guidance for planning software builds and releases.

Scope

This guideline should be used by all software projects and should align with the Software Project Planning process.

Guideline

The following are in no particular order and are numbered for reference purposes only:

1. Understand the Differences Between Builds and Releases

A software build is a portion of a system that satisfies an identifiable subset of the total software requirements. As the software is developed, each successive software build includes the functionality allocated to that build, in addition to the functionality included in preceding builds.

A Release is a Build that is delivered to the Customer for formal testing.

Apply the build guidelines to both builds and releases.

2. Apply an Incremental Software Development Approach

Develop the software incrementally, in software builds and releases, so that requirements misunderstandings, design flaws, implementation errors, and productivity problems can be detected and resolved early in the software development life cycle.

The software build approach is an incremental approach to implementing, integrating, and testing software systems. It reduces potential problems in software development to manageable levels that can be effectively addressed with clear corrective actions. Each successive software build adds increased functionality to a proven baseline, thereby reducing schedule risk and increasing schedule credibility. The key to the software build approach is in planning how to partition and sequence the incremental development of the software that comprises each software build.

3. Develop a Build/Release Plan

Define the build contents and the implementation schedule for each build in a build plan within your Software Management Plan/Product Plan (SMP/PP).

The build/release contents are typically driven by defining the new or upgraded functionality that will be included in the build/release (see item

4 below). For systems being maintained, build/release contents are typically driven by the set of changes implemented in response to change requests and/or noncompliance’s that will be included in the build/release (see item 6 below).

Update the Build Plan at the end of each build as well as whenever the Software Management Plan is updated. Review the Build Plan with your management.

4. Define the Contents of Each Build/Release

When planning the software development project’s build/release contents, consider the following guidelines:

- Plan that each build/release provide complete and demonstrable functions that add to the functions of predecessor builds/releases
- Keep the first build/release as simple as possible especially if the staff is not already familiar with the application, computer, development environment, operating system, language, standards, procedures, or external interfaces.
- The first build should consist of a basic foundation for the system software to provide a functionally meaningful subset of the final software capability. It should provide a complete and definitive support structure for the system software, providing a foundation or framework upon which additional functionality can later be added in subsequent builds.
- Work around long lead times for procurement or development of acquired hardware, software, or firmware components by planning to use substitute components as much as possible. This would include the use of a development computer during early builds while waiting for the target host computer, or the simulation of a hardware interface in software until the real interface is available.
- Do not postpone to the last build capabilities crucial to the operational use of the software. For example, do not postpone capabilities critical to system usability, stability, or performance to the last build.
- Plan capabilities that address high-risk requirements or complex requirements or that can have a major impact on the design for earlier builds/releases so that problems can be resolved early.
- Delay capabilities whose requirements are incomplete to later builds/releases to allow time to more clearly solidify the requirements.
- Plan to include capabilities critical to the usability, stability, or performance of the software in the next-to-last build
- Plan a “clean-up” build/release as the last build/release to resolve change requests or problem reports the customer deems necessary before final delivery of the system.

5. Schedule the Builds/Releases

A build implementation effort, including unit design, code, test, and integration, should span 3 to 5 months. Add at least 1 more month for Build Testing by an independent test team. If a software build is to be integrated into a system build with other developed or acquired system elements, add more time for system integration and testing. Consult with the team performing the integration (who should involve stakeholders from each subsystem) to determine the complexity of the integration and testing and determine how much time should be scheduled. Separately schedule the Formal System Testing for each release.

Do not overlap the development efforts for multiple builds; this is an approach that increases CM and testing complexity (and may require a fix to be applied and tested to multiple baselines). At most, plan the Build Testing for one build in parallel with development of the next build, if there is an independent test team.

If schedule constraints require that implementation efforts overlap for multiple builds, divide the software into multiple components that have a few, well-defined interfaces. Assign each component to a separate project group and implement the components in parallel. Define a build plan for each component.

Synchronize the build schedules so that a build for each component is released for qualification testing at approximately the same time. The effectiveness of qualification testing often depends on the stability of the software being tested.

Do not schedule a programmer to work on two builds simultaneously. This seldom works because some programmers may decide to work hardest on the build that contains the most interesting software instead of giving top priority to the software in the earlier of the two builds. However, this rule does not apply to the efforts needed to correct defects in software released for qualification testing. A programmer who releases defective software is still responsible for correcting the defects even if it means interrupting the implementation efforts for the next build.

Consider any Customer needs for specific operational capabilities as you develop the schedule.

It is recommended that the schedule for and contents of each build/release be reviewed with and concurred by all relevant stakeholders including the customer and line management at the Branch Status Review for the project (and other forums for those stakeholders not present).

6. Maintenance Build/Release Considerations

Maintenance efforts are often measured in terms of days or weeks instead of months. However, control will be lost over maintenance if each maintenance item is planned and managed as a separate item. Instead, group maintenance items into maintenance releases, analogous to builds. Establish a regular schedule for maintenance releases (e.g., one every 4 or 6 months). Do not agree to an endless series of emergency releases, as adequate planning will not be possible and control will soon be lost.

Recognize that emergencies will arise. Non-conformances that are critical to continued operations will require an emergency release.

When planning the software maintenance project's build/release contents, try to: • Plan for resolution of the highest-priority change requests and/or non-conformances in the upcoming build/release • Include, as appropriate, resolution of lower-priority change requests and/or non-conformances that can be addressed in the same areas of the system as higher-priority changes that are being implemented.

Tools and Templates

The following tools are available. Others may exist for the local project.

Name	Description
None identified	

Change History

Version	Date	Description of Improvements
1.0	9/19/09	Initial version approved by CCB